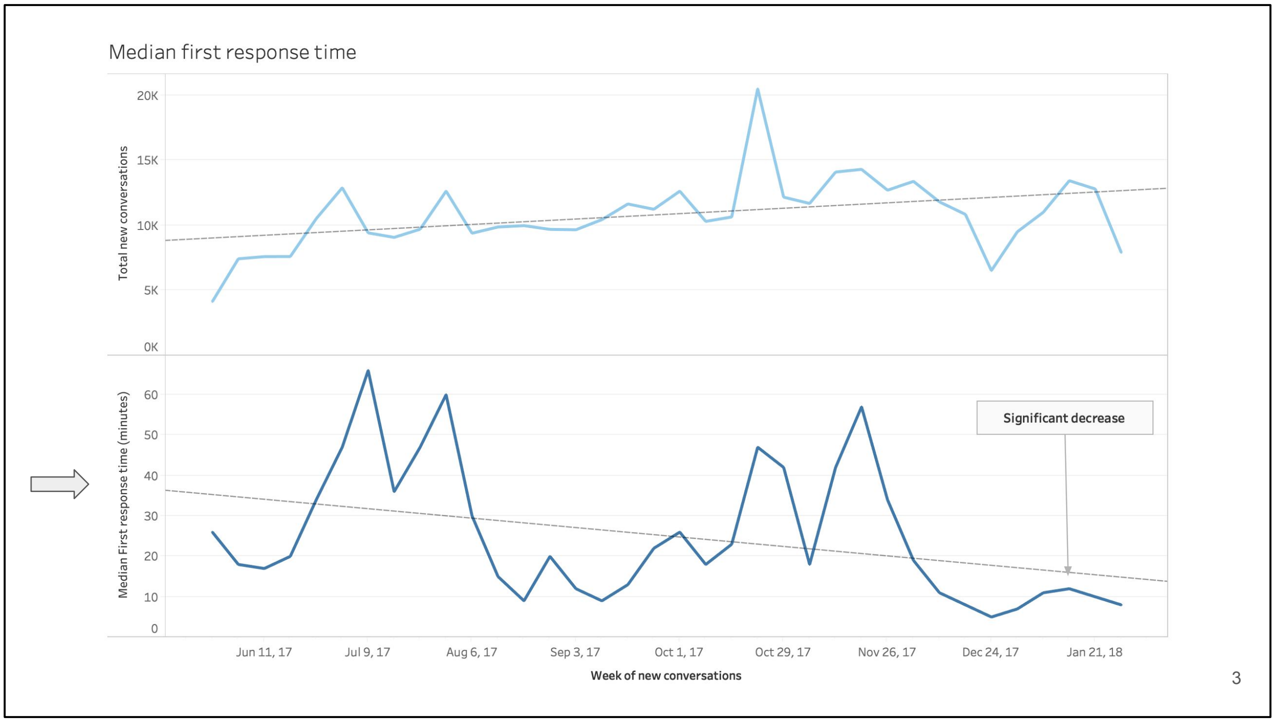
# Customer Operations Assignment

Diego Alonso

# Overview

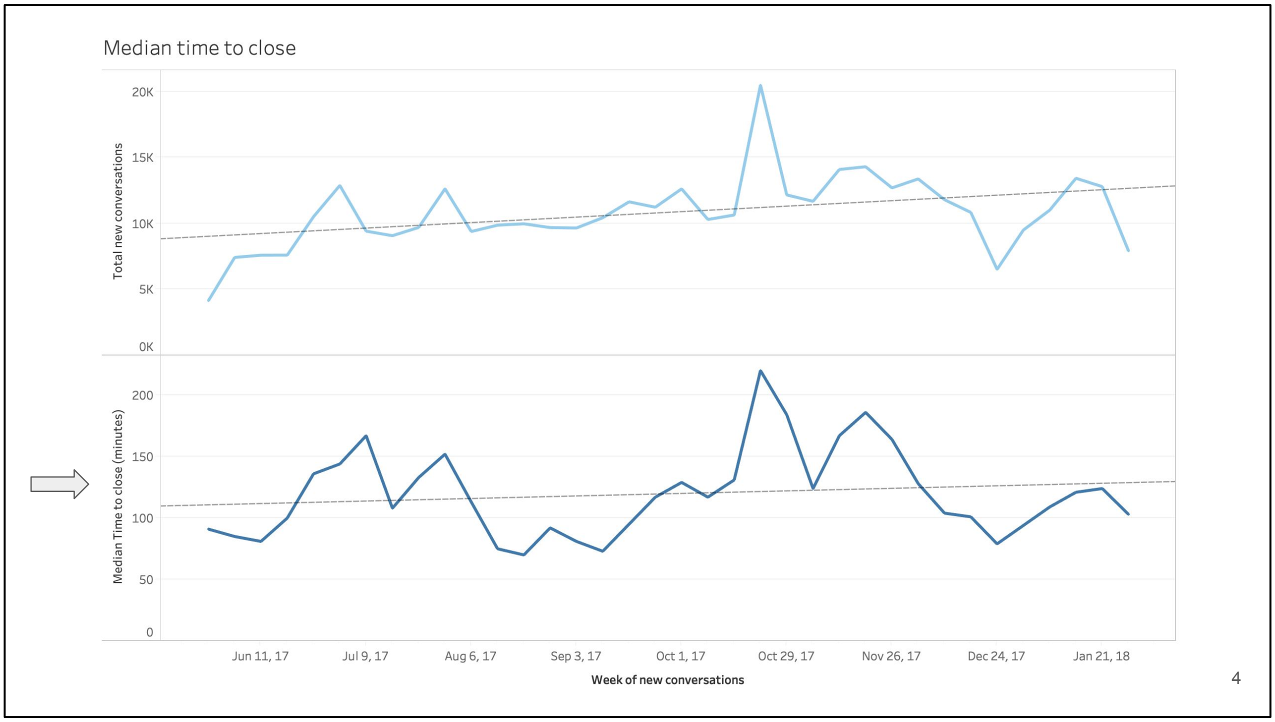
- Topics
  - Analyze productivity of the whole customer support team
  - Analyze productivity on an individual agent level
  - Segment agents by productivity
- Appendix. Understanding the data




One way to measure productivity on the organization level is to analyze the **median first response time** to a conversation and also the **median time to close**.

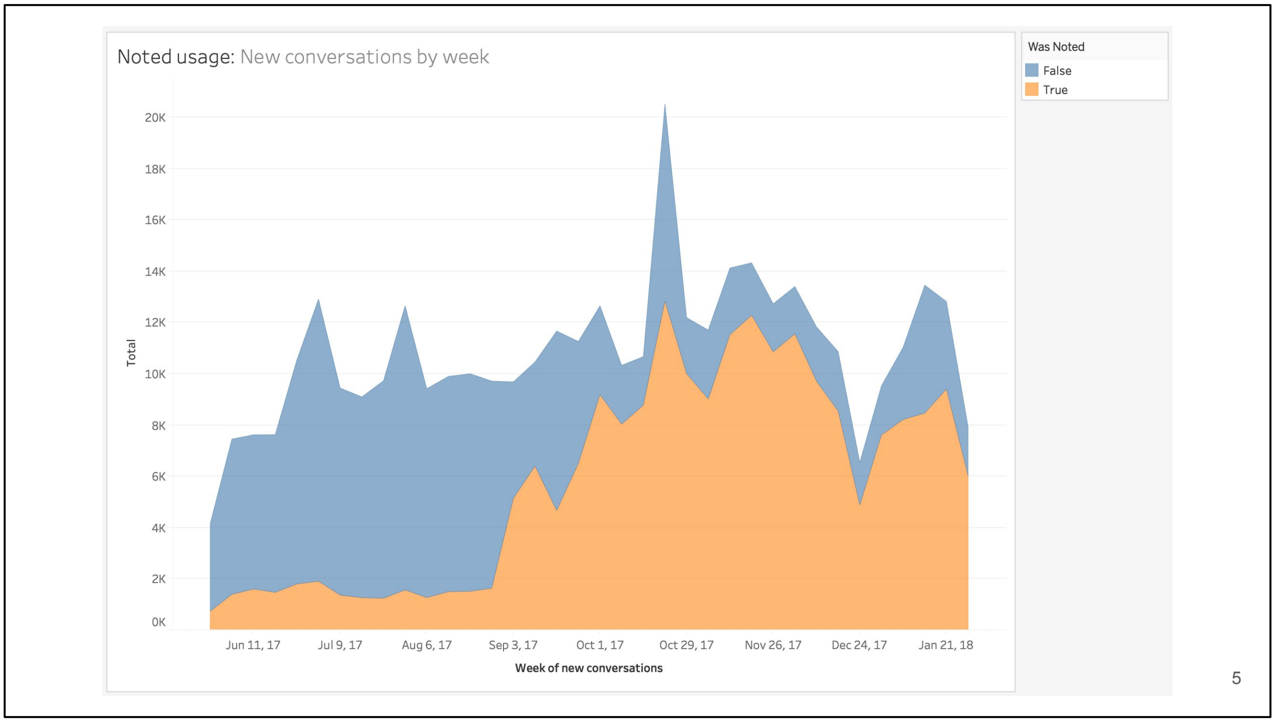
We can see that the organization as a whole has improved its median first response time with a **significant decrease** a the end of our analysis period.

Also as a whole even though the conversation volume is growing the median first response time is decreasing as we can see on the **trend line** below.




When we look at **median time to close**, it’s interesting how we don’t see a significant decrease at the end of the year or a decreasing trend line as we saw on median first response time.

In this case it would be worth to understand **what does closing a conversation mean** as a process. It seems that the team are **optimizing for median first response time** to provide a better customer experience.

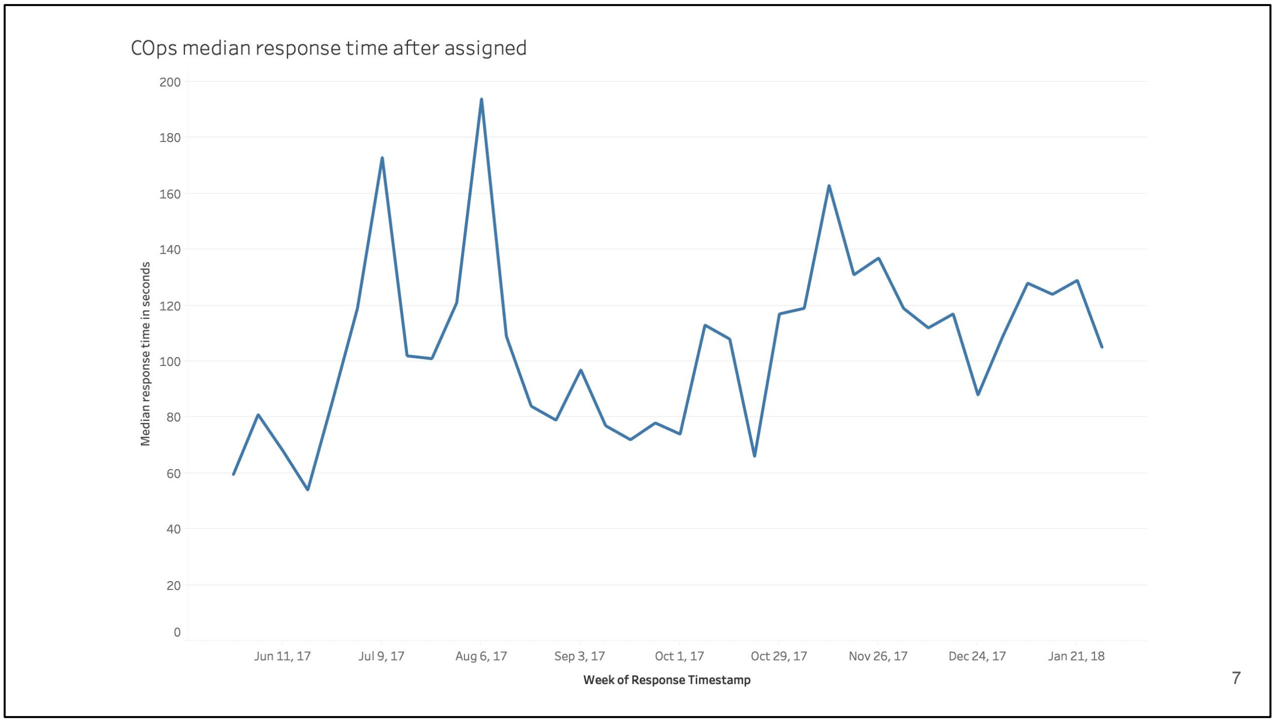



After analyzing event types present in a conversation we see a significant increase in the usage of the **Noted** feature. Over time, the team could have gotten more familiar or have found a way to leverage this feature to improve first response time by having noted a conversation.

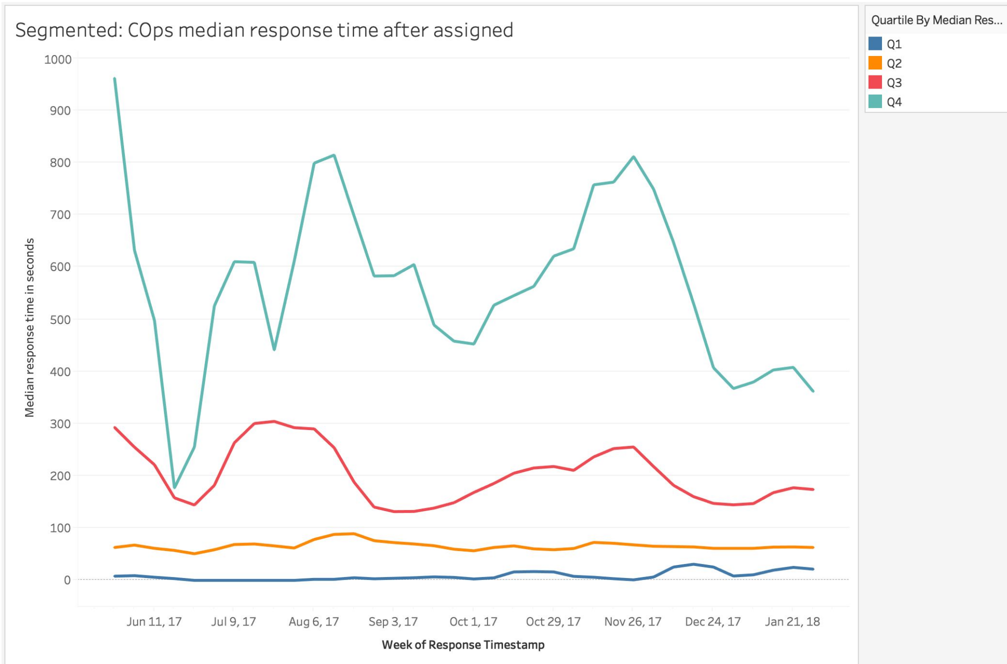
# Creating target metrics

The customer operations team can also set different SLAs to keep tracking and improving their performance over time.

For example, there could be a **target on first response time** where we would like to keep the **P90 first response** time below a certain value.



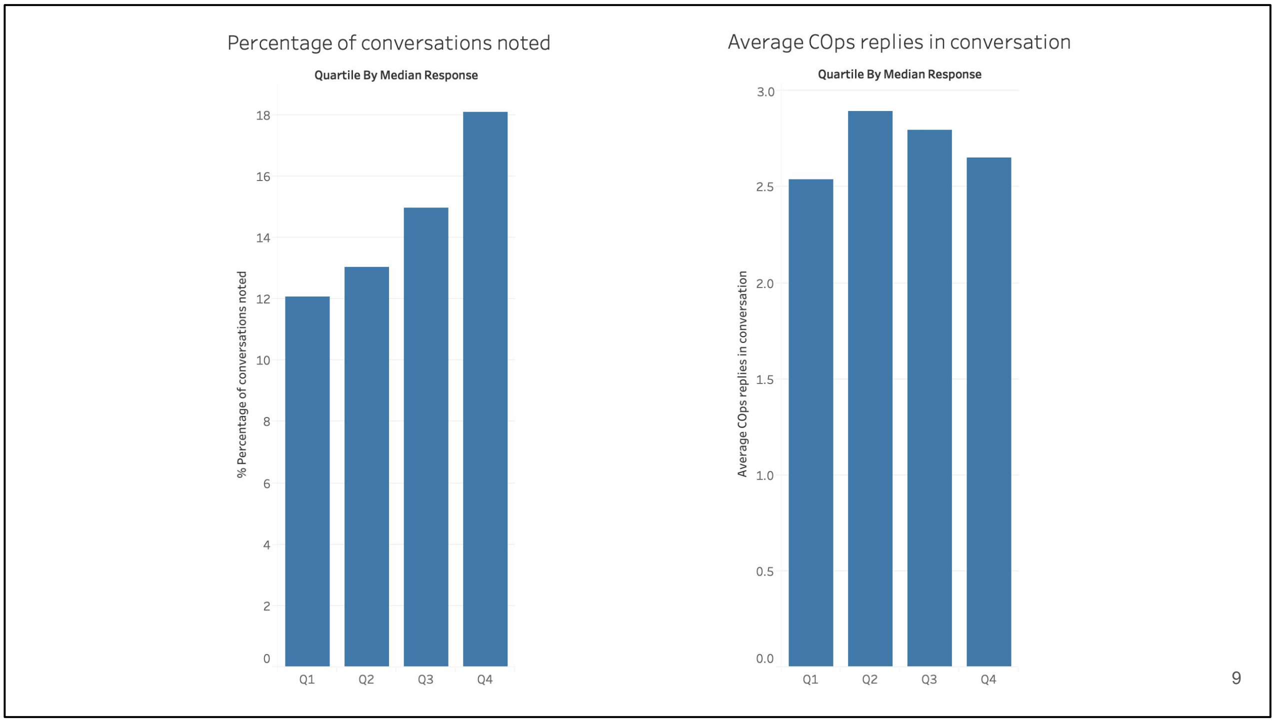

One way to measure productivity on a customer operations **agent level** is by analyzing **median response time after a conversation has been assigned**.






We can segment customer support agents by **quartiles** using **median response** time after assigned.

- **Q1** < 25th percentile
- **Q2** < 50th percentile
- **Q3** < 75th percentile
- **Q4** < 100th percentile




When we analyze the conversations in which the admins of each quartiles participated, one thing strikes as a factor for performance.

It seems that admins in the lower performing quartiles **(higher median response time)** participated in **more conversations that had noted events**. One hypothesis to performance is that these conversation by nature are less straightforward to follow up with than the ones that were annotated.

The **average number of admin replies per conversation** is lower for the highest performing quartile (lowest median response time), but not far from the lowest performing one, so we can't venture to make any hard claims here.

## Cohort by start month: Monthly median response time

| Month of Cohort | June 2017 | July 2017 | August 2017 | September 2017 | October 2017 | November 2017 | December 2017 | January 2018 |
|---|---|---|---|---|---|---|---|---|
| June 2017 | 69.0 | 121.0 | 101.0 | 77.0 | 61.0 | 90.0 | 91.0 | 90.0 |
| July 2017 | | 99.0 | 158.0 | 73.0 | 52.0 | 68.0 | 47.0 | 58.0 |
| August 2017 | | | 92.0 | 91.0 | 87.0 | 83.0 | 95.0 | 117.0 |
| September 2017 | | | | 62.0 | 117.0 | 192.0 | 121.0 | 148.0 |
| October 2017 | | | | | 171.0 | 231.0 | 160.0 | 147.0 |
| November 2017 | | | | | | 156.5 | 115.0 | 127.0 |
| December 2017 | | | | | | | 102.0 | 103.0 |
| January 2018 | | | | | | | | 127.0 |

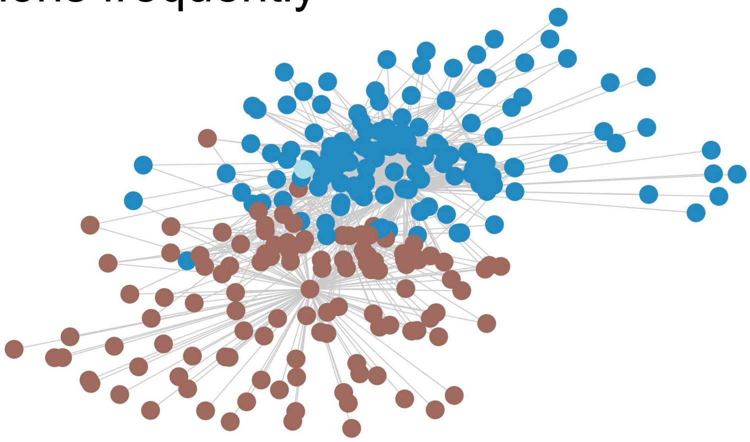


We can also segment agents in cohorts using their start month to see if they improve their performance over time.

The **best performing cohort** is the one of employees that joined in July 2017.

And the **lowest performing cohort** is the one of October 2017. We remember from our first slides that there was a spike of new conversations started in October 2017 related to current accounts rollout. So it's possible that this group had to join the front-lines right away without **receiving proper training**.

## Inferring teams from agents participating in the same conversations frequently





And finally we can also segment agents by trying to infer their team.

We created a graph to relate admins that participate in the same conversations. There’s a good chance that customer operations agents that participate in the same conversations a high number of times could **belong to the same team like Fraud agents or Product Support agents**.

I did some quick explorations tweaking the weight on the edges to see if we could find an evident groups using Markov Clustering but did not find anything significant on a quick pass.

# Thanks for reading

Don’t forget to review the following section

# Understanding the data

## Segmenting customer support team by creating a graph of agents that participated in the same conversation

```
SELECT conversation_id, deduped_agent_id
FROM `analytics-take-home-test.take_home_DA.deduped_events`
GROUP BY 1, 2
ORDER BY 1
```

- We'll create a graph to relate admins that participate in the same conversations. There's a good chance that agents that participate in the same conversations a high number of times could belong to the same team like Fraud agents or Product Support agents.

### Creating the graph edges and weighting them by number of conversations in common.

**generate_agents_graph.rb**

```
require 'csv'
require 'pp'

def link_nodes(network, nodes)
  new_connections = {}
  nodes.each do |node_id|
    new_connections[node_id] = Hash[nodes.select{|id| id != node_id}.collect{|id| [id, 1]}]
  end
  new_connections.each do |node_id, new_nodes|
    if network.key?(node_id)
      new_nodes.each do |id, count|
        if network[node_id].key?(id)
          network[node_id][id] += 1
        else
          network[node_id][id] = 1
        end
      end
    else
      network[node_id] = new_nodes
    end
  end
end

agents_network = {}
count = 0
cur_conversation = 0
convo_agents = []
CSV.foreach('conversations_agents.csv', :headers => true) do |row|
```

```
  if cur_conversation != row[0]
    cur_conversation = row[0]
    link_nodes(agents_network, convo_agents)
    convo_agents = []
    count += 1
  end
  convo_agents << row[1]
end

output = CSV.new(File.open('agents_graph.csv', 'w'))
agents_network.each do |agent_id, connections|
  connections.each do |other_agent_id, count|
    output << [agent_id, other_agent_id, count]
  end
end
```

- Initially I coded a quick script to create the weighted graph edges in order to try to draw in Tableau, but in the end I had a better result drawing the graph in Python.

**draw_agents_graph.py**

```
import csv
import networkx as nx
import matplotlib.pyplot as plt
import markov_clustering as mc

skip_list = []

G = nx.Graph()

with open('agents_graph.csv', mode='r') as csv_file:
  csv_reader = csv.reader(csv_file, delimiter=',')
  line_count = 0
  for row in csv_reader:
    line_count += 1
    node = row[0]
    related_node = row[1]
    if node in skip_list or related_node in skip_list:
      next
    related_count = int(row[2])
    # if related_count > 100 and related_count < 1000:
    G.add_edge(node, related_node, weight=related_count)

  print(f'Processed {line_count} lines.')


print("Number of nodes:", G.number_of_nodes())
print("Number of edges:", G.number_of_edges())

# nx.draw(G, with_labels=False)
# plt.show()

matrix = nx.to_scipy_sparse_matrix(G)
result = mc.run_mcl(matrix)
clusters = mc.get_clusters(result)
mc.draw_graph(matrix, clusters, with_labels=False, edge_color="silver")
```

```
plt.show()
```

- I did some quick explorations tweaking the weight on the edges to see if we could find an evident groups using Markov Clustering but did not find anything significant on a first pass.